**Web Scraper & Data Organizer - Technical Documentation**

**Project Overview**

A comprehensive web scraping solution designed to save thousands of HTML pages with all associated assets (images, CSS, JavaScript) while organizing them into specified folder structures. The system includes bot detection bypass mechanisms and supports both spreadsheet-based batch processing and API-driven individual page processing.

**Core Requirements Analysis**

**Functional Requirements**

* **Batch Processing**: Handle 1000+ HTML pages efficiently
* **Complete Asset Preservation**: Download and organize images, CSS, JavaScript files
* **Flexible Input Methods**:
  + Spreadsheet upload with URLs and save paths
  + API interface for individual page processing
* **Custom Organization**: User-defined folder structures
* **Bot Detection Bypass**: Implement strategies to avoid detection and blocking

**Non-Functional Requirements**

* **Performance**: Handle large volumes without system overload
* **Reliability**: Retry mechanisms for failed downloads
* **Scalability**: Queue-based processing for concurrent operations
* **User Experience**: Intuitive interface with progress tracking

**System Architecture**

**Technology Stack**

* **Backend**: Python with FastAPI framework
* **Web Scraping**: Selenium WebDriver + Beautiful Soup 4
* **Database**: PostgreSQL for job tracking, user management, and metadata
* **Queue System**: Redis with Celery for async processing
* **Frontend**: React.js with TypeScript
* **Authentication**: JWT tokens with refresh mechanism
* **File Storage**: Local filesystem with user-specific directories
* **Security**: bcrypt for password hashing, rate limiting

**Core Components**

**1. Authentication System**

* **User Registration**: Email verification, password strength validation
* **Login System**: JWT token-based authentication with refresh tokens
* **Session Management**: Secure token storage and automatic renewal
* **Password Security**: bcrypt hashing with salt rounds
* **Rate Limiting**: Login attempt throttling and account lockout

**2. Web Scraper Engine**

# Core scraping functionality

- Selenium WebDriver for dynamic content

- Beautiful Soup for HTML parsing

- Asset discovery and download

- Bot detection bypass mechanisms

- User-specific job isolation

**3. Bot Bypass Module**

* **User Agent Rotation**: Dynamic browser fingerprinting
* **Request Throttling**: Configurable delays between requests
* **Proxy Support**: Rotating proxy pools
* **Session Management**: Cookie and session persistence
* **CAPTCHA Handling**: Integration with solving services
* **Stealth Mode**: Browser automation detection evasion

**4. Asset Management System**

* **Asset Discovery**: CSS, JS, image URL extraction
* **Dependency Mapping**: Maintain relative path relationships
* **URL Rewriting**: Convert absolute to relative paths
* **Duplicate Detection**: Avoid redundant downloads

**5. Data Organization Engine**

* **Folder Structure Management**: Custom path creation
* **File Naming Conventions**: Configurable naming schemes
* **Metadata Storage**: Track download history and status
* **Conflict Resolution**: Handle duplicate filenames

**Database Schema**

**Tables Structure**

**users**

CREATE TABLE users (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

email VARCHAR(255) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

is\_active BOOLEAN DEFAULT true,

is\_verified BOOLEAN DEFAULT false,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

last\_login TIMESTAMP,

subscription\_plan ENUM('free', 'pro', 'enterprise') DEFAULT 'free'

);

**user\_sessions**

CREATE TABLE user\_sessions (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id) ON DELETE CASCADE,

refresh\_token VARCHAR(512) NOT NULL,

expires\_at TIMESTAMP NOT NULL,

created\_at TIMESTAMP DEFAULT NOW(),

last\_used TIMESTAMP DEFAULT NOW(),

ip\_address INET,

user\_agent TEXT

);

**jobs**

CREATE TABLE jobs (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id) ON DELETE CASCADE,

name VARCHAR(255) NOT NULL,

status ENUM('pending', 'running', 'completed', 'failed', 'paused'),

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

total\_pages INTEGER,

completed\_pages INTEGER DEFAULT 0,

failed\_pages INTEGER DEFAULT 0,

settings JSONB,

output\_directory TEXT NOT NULL

);

**pages**

CREATE TABLE pages (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

job\_id UUID REFERENCES jobs(id) ON DELETE CASCADE,

url TEXT NOT NULL,

save\_path TEXT NOT NULL,

status ENUM('pending', 'downloading', 'completed', 'failed'),

created\_at TIMESTAMP DEFAULT NOW(),

completed\_at TIMESTAMP,

error\_message TEXT,

file\_size BIGINT,

asset\_count INTEGER DEFAULT 0

);

**assets**

CREATE TABLE assets (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

page\_id UUID REFERENCES pages(id) ON DELETE CASCADE,

url TEXT NOT NULL,

local\_path TEXT NOT NULL,

asset\_type ENUM('css', 'js', 'image', 'font', 'other'),

file\_size BIGINT,

downloaded\_at TIMESTAMP DEFAULT NOW()

);

**API Specifications**

**REST Endpoints**

**Authentication**

POST /api/v1/auth/register # User registration

POST /api/v1/auth/login # User login

POST /api/v1/auth/refresh # Refresh JWT token

POST /api/v1/auth/logout # User logout

POST /api/v1/auth/forgot-password # Password reset request

POST /api/v1/auth/reset-password # Password reset confirmation

GET /api/v1/auth/verify-email # Email verification

GET /api/v1/auth/me # Get current user profile

PUT /api/v1/auth/profile # Update user profile

**Job Management**

POST /api/v1/jobs # Create new scraping job

GET /api/v1/jobs # List all jobs

GET /api/v1/jobs/{id} # Get job details

DELETE /api/v1/jobs/{id} # Cancel/delete job

POST /api/v1/jobs/{id}/start # Start job execution

POST /api/v1/jobs/{id}/pause # Pause job execution

**Single Page Processing**

POST /api/v1/pages/scrape # Scrape single page

GET /api/v1/pages/{id} # Get page status

**File Upload**

POST /api/v1/upload/spreadsheet # Upload CSV/Excel with URLs

**Request/Response Examples**

**User Registration**

POST /api/v1/auth/register

{

"email": "user@example.com",

"password": "SecurePass123!",

"first\_name": "John",

"last\_name": "Doe",

"confirm\_password": "SecurePass123!"

}

Response:

{

"message": "Registration successful. Please check your email for verification.",

"user\_id": "550e8400-e29b-41d4-a716-446655440000"

}

**User Login**

POST /api/v1/auth/login

{

"email": "user@example.com",

"password": "SecurePass123!"

}

Response:

{

"access\_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",

"refresh\_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",

"user": {

"id": "550e8400-e29b-41d4-a716-446655440000",

"email": "user@example.com",

"first\_name": "John",

"last\_name": "Doe",

"subscription\_plan": "free"

}

}

**Create Job**

POST /api/v1/jobs

{

"name": "Website Backup - Product Pages",

"pages": [

{

"url": "https://example.com/product/1",

"save\_path": "/backups/products/product-1"

}

],

"settings": {

"delay\_between\_requests": 2,

"use\_proxy": true,

"max\_retries": 3,

"asset\_types": ["css", "js", "images"]

}

}

**Bot Detection Bypass Strategies**

**1. Browser Fingerprinting Evasion**

* Dynamic User-Agent strings
* Randomized viewport sizes
* Browser plugin simulation
* Canvas fingerprint randomization

**2. Behavioral Patterns**

* Human-like scrolling patterns
* Random mouse movements
* Variable typing speeds
* Realistic pause durations

**3. Network-Level Evasion**

* Residential proxy rotation
* DNS over HTTPS
* TLS fingerprint randomization
* Connection timing variation

**4. Rate Limiting Compliance**

* Intelligent backoff algorithms
* Respect robots.txt (when configured)
* Monitor response codes for throttling
* Adaptive delay adjustments

**File Organization Strategy**

**Directory Structure**

/output/

├── jobs/

│ └── {job\_id}/

│ ├── pages/

│ │ └── {domain}/

│ │ └── {path\_structure}/

│ │ ├── index.html

│ │ └── assets/

│ │ ├── css/

│ │ ├── js/

│ │ └── images/

│ └── metadata/

│ ├── job\_info.json

│ └── page\_mapping.json

**Asset Path Rewriting**

* Convert absolute URLs to relative paths
* Maintain original folder structure when possible
* Handle duplicate asset names with hash suffixes
* Create asset manifest for each page

**Performance Optimization**

**Concurrent Processing**

* Thread pool for I/O operations
* Process pool for CPU-intensive tasks
* Queue-based job distribution
* Resource-aware throttling

**Memory Management**

* Streaming downloads for large files
* Periodic garbage collection
* Asset deduplication
* Efficient HTML parsing

**Monitoring & Logging**

* Real-time progress tracking
* Error logging with stack traces
* Performance metrics collection
* Resource utilization monitoring

**Security Considerations**

**Data Protection**

**Data Protection**

* **User Isolation**: Strict data separation between users
* **Input Validation**: Comprehensive sanitization and validation
* **SQL Injection Prevention**: Parameterized queries and ORM usage
* **XSS Protection**: Content Security Policy and input sanitization
* **File Access Control**: User-specific directory permissions
* **HTTPS Enforcement**: All communication encrypted in transit

**Network Security**

* Proxy credential encryption
* SSL certificate verification
* Request header sanitization
* IP address whitelisting options

**Deployment Architecture**

**Development Environment**

services:

app:

build: .

ports: ["8000:8000"]

depends\_on: [db, redis]

db:

image: postgres:14

environment:

POSTGRES\_DB: webscraper

POSTGRES\_USER: scraper

POSTGRES\_PASSWORD: secure\_password

redis:

image: redis:7-alpine

worker:

build: .

command: celery worker

depends\_on: [db, redis]

**Production Considerations**

* Load balancer for API endpoints
* Database connection pooling
* Redis cluster for high availability
* File storage backup strategy
* Monitoring with Prometheus/Grafana

**Error Handling & Recovery**

**Retry Mechanisms**

* Exponential backoff for network errors
* Different retry strategies per error type
* Maximum retry limits per page
* Circuit breaker pattern for failing domains

**Failure Recovery**

* Job state persistence
* Resumable downloads
* Partial success handling
* Error reporting and notifications

**Configuration Management**

**Environment Variables**

# Database

DATABASE\_URL=postgresql://user:pass@localhost/webscraper

REDIS\_URL=redis://localhost:6379

# Authentication

JWT\_SECRET\_KEY=your-super-secret-jwt-key-here

JWT\_ACCESS\_TOKEN\_EXPIRE\_MINUTES=15

JWT\_REFRESH\_TOKEN\_EXPIRE\_DAYS=30

# Application

OUTPUT\_DIRECTORY=/var/www/scraped\_content

MAX\_CONCURRENT\_JOBS\_PER\_USER=3

GLOBAL\_MAX\_CONCURRENT\_JOBS=10

# Email Service

SMTP\_HOST=smtp.gmail.com

SMTP\_PORT=587

SMTP\_USERNAME=your-email@gmail.com

SMTP\_PASSWORD=your-app-password

# Scraping

DEFAULT\_USER\_AGENT=Mozilla/5.0...

PROXY\_POOL\_URL=http://proxy-service/api/v1/proxies

CAPTCHA\_SOLVER\_API\_KEY=your-2captcha-api-key

**Runtime Configuration**

* Per-job scraping parameters
* Global rate limiting settings
* Asset type filtering
* Custom header injection

**Testing Strategy**

**Unit Tests**

* Individual component testing
* Mock external dependencies
* Edge case validation
* Performance benchmarking

**Integration Tests**

* End-to-end workflow testing
* Database transaction testing
* API endpoint validation
* File system operation testing

**Load Testing**

* Concurrent job processing
* High-volume page scraping
* Resource utilization under load
* Failure scenario testing

**Maintenance & Operations**

**Monitoring**

* Job success/failure rates
* Average processing time per page
* System resource utilization
* Error pattern analysis

**Maintenance Tasks**

* Database cleanup routines
* Log rotation and archival
* Temporary file cleanup
* Performance optimization reviews

**UI/UX Design Specifications**

**Authentication Pages**

**Login Page**

* **Layout**: Centered form with company branding
* **Fields**: Email, password, "Remember me" checkbox
* **Features**:
  + Password visibility toggle
  + "Forgot password?" link
  + Social login options (Google, GitHub)
  + Registration redirect link
* **Validation**: Real-time form validation with error messages
* **Security**: Rate limiting display, CAPTCHA after failed attempts

**Registration Page**

* **Layout**: Multi-step form or single comprehensive form
* **Fields**: First name, last name, email, password, confirm password
* **Features**:
  + Password strength indicator
  + Terms of service acceptance
  + Email verification flow
  + Account activation process
* **Validation**: Real-time password matching, email format validation

**Password Reset Flow**

* **Forgot Password**: Email input with verification
* **Reset Email**: Secure token-based reset link
* **New Password**: Password reset form with confirmation
* **Success**: Confirmation and automatic login redirect

**Main Application Interface**

**Dashboard Layout**

* **Header**: Navigation, user menu, notifications
* **Sidebar**: Main navigation (Dashboard, Jobs, Settings, Billing)
* **Main Content**: Job overview, statistics, recent activity
* **Footer**: Support links, version info

**Job Management Interface**

* **Job List**: Sortable table with status, progress, actions
* **Job Details**: Expandable rows with detailed information
* **Quick Actions**: Start, pause, stop, clone, delete buttons
* **Bulk Operations**: Select multiple jobs for batch actions

**Job Creation Wizard**

* **Step 1**: Basic information (name, description)
* **Step 2**: URL input (manual entry or file upload)
* **Step 3**: Output settings (directory, naming conventions)
* **Step 4**: Scraping configuration (delays, assets, proxy settings)
* **Step 5**: Review and confirmation

**Real-time Monitoring**

* **Progress Bars**: Visual progress indicators for each job
* **Live Logs**: Scrollable log viewer with filtering
* **Status Updates**: WebSocket-based real-time status updates
* **Error Notifications**: Toast notifications for failures

**Responsive Design**

* **Mobile-First**: Optimized for mobile devices
* **Tablet Support**: Adapted layouts for tablet screens
* **Desktop Enhancement**: Full-featured desktop experience
* **Touch-Friendly**: Large buttons and touch targets

**Accessibility Features**

* **WCAG 2.1 AA Compliance**: Full accessibility standard compliance
* **Keyboard Navigation**: Complete keyboard accessibility
* **Screen Reader Support**: ARIA labels and semantic HTML
* **High Contrast Mode**: Support for accessibility themes
* **Focus Indicators**: Clear focus states for all interactive elements

**User Experience Enhancements**

* **Loading States**: Skeleton screens and progressive loading
* **Error Handling**: Graceful error messages and recovery options
* **Onboarding**: Interactive tutorial for new users
* **Contextual Help**: Tooltips and help text throughout the interface
* **Keyboard Shortcuts**: Power user keyboard shortcuts

**Future Enhancements**

**Phase 2 Features**

* **Multi-tenant Architecture**: Enterprise team management
* **Advanced Scheduling**: Cron-based job scheduling
* **Machine Learning Integration**: Intelligent bot detection bypass
* **Advanced Analytics**: Detailed scraping performance analytics
* **API Rate Limiting**: Per-user API quotas and throttling
* **Webhook Notifications**: External integrations for job completion

**Phase 3 Features**

* **Cloud Storage Integration**: AWS S3, Google Cloud Storage
* **Advanced User Management**: Role-based access control (RBAC)
* **Billing Integration**: Subscription management with Stripe
* **Advanced Proxy Management**: Residential proxy pool management
* **Custom Browser Extensions**: Browser extension for easy URL collection
* **Mobile Applications**: Native iOS and Android apps

**Scalability Roadmap**

* **Microservices Architecture**: Service decomposition for scale
* **Kubernetes Deployment**: Container orchestration for production
* **Global CDN Integration**: Asset delivery optimization
* **Database Sharding**: Horizontal database scaling
* **Load Balancing**: Multi-region deployment strategy
* **Monitoring & Observability**: Comprehensive system monitoring